Assignment7

Jnaneswara Rao Rompilli [EE20B052]

April 8, 2022

1 Introduction

This assignemnt involves the analysis of filters using Laplace Transforms. Python's symbolic solving library, Sympy is used to solve Modified Nodal Analysis equations. Scipy library is used to calculate output responses of the systems

Low Pass Filter

The low pass filter that we use gives the following matrix equation after simplification of the modified nodal equations.

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{1}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1} - \frac{1}{R_2} - sC_1 & \frac{1}{R_2} & 0 & sC_1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)/R_1 \end{pmatrix}$$

Below is the code for Low Pass Filter

```
def lowpass(R1, R2, C1, C2, G, Vi):
    A = Matrix(
        [   [0, 0, 1, -1 / G], [-1 / (1 + s * R2 * C2), 1, 0, 0],
            [0, -G, G, 1], [-1 / R1 - 1 / R2 - s * C1, 1 / R2, 0, s * C1], ]
    )
    b = Matrix([0, 0, 0, -Vi / R1])
    V = A.inv() * b  # [V1 Vp Vm Vo]
    return (A, b, V)
```

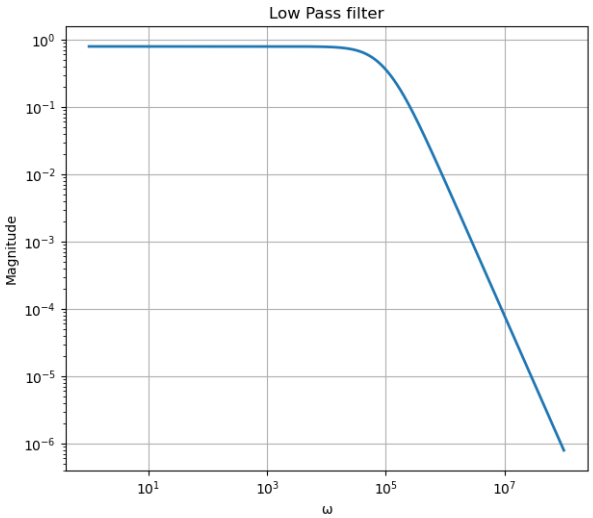

Figure 1: Magnitude response of Low Pass filter

High Pass Filter

The high pass filter we use gives the following matrix equations after simplification of the modified nodal equations

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{-sR_3C_2}{1+sR_3C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -1-(sR_1C_1)-(sR_3C_2)) & sC_2R_1 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)sR_1C_1 \end{pmatrix}$$

Below is the code for High Pass Filter

```
def highpass(R1, R3, C1, C2, G, Vi):
    A = Matrix(
        [
            [0, -1, 0, 1 / G],[s * C2 * R3 / (s * C2 * R3 + 1), 0, -1, 0],
            [0, G, -G, 1],[-s * C2 - 1 / R1 - s * C1, 0, s * C2, 1 / R1],
        ]
    )
    b = Matrix([0, 0, 0, -Vi * s * C1])
    V = A.inv() * b
    return (A, b, V)
```

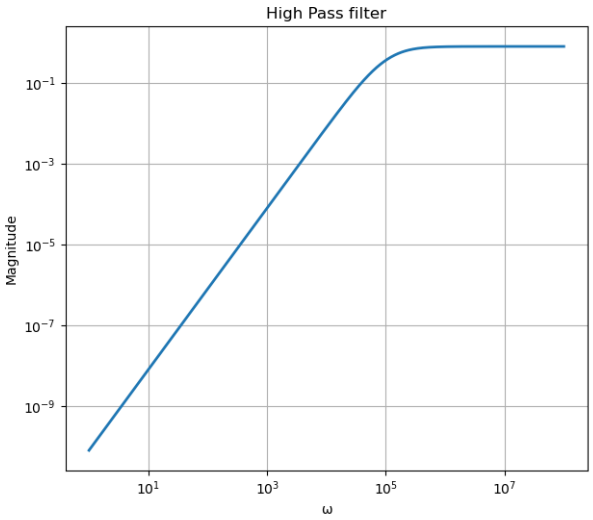

Figure 2: Magnitude response of High Pass filter

2 Tasks

2.1 Step response Low Pass Filter

Output Voltage for unit step function as an input

```
# Step response of a Low pass filter
A1, b1, H1 = lowpass(10000, 10000, 1e-9, 1e-9, 1.586, 1 / s)
Vo = H1[3]
Vh = sympytolti(Vo)

t, Vt = sp.impulse(Vh, None, np.linspace(0, 5e-3, 10000))
pylab.figure(2, figsize=(7, 6))
pylab.plot(t, Vt, label="V(t)")
pylab.title("Step response of Low Pass filter", fontsize=12)
pylab.xlabel("t", fontsize=10)
pylab.ylabel("Vo(t)", fontsize=10)
pylab.grid()
pylab.legend()
```

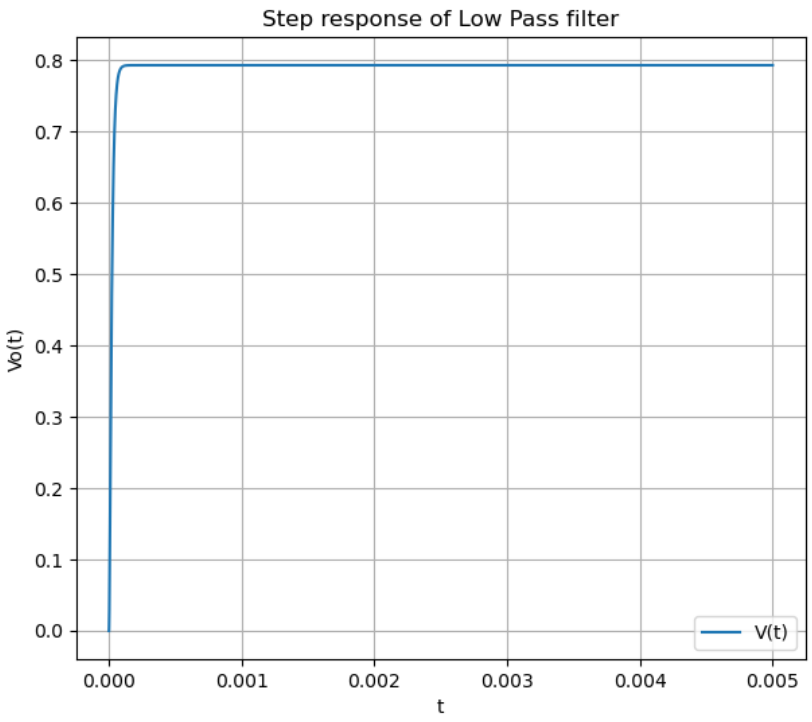

Figure 3: Step response of Low Pass filter

2.2 Response to sum of sinusoids

Input function is,

$$V_i(t) = (\ \sin(2000\pi t) + \cos\left(2 * 10^6 \pi t\right)\ ) u_o(t)\ Volts$$

Output response for the Low Pass filter can be found as shown in the code snippet.

```
Vit = np.sin(2000 * PI * t) + np.cos(2e6 * PI * t)
t, Vo, svec = sp.lsim(H, Vit, t)

pylab.figure(3, figsize=(7, 6))
pylab.plot(t, Vo)
pylab.title("Output voltage Vo(t) (Low pass)", fontsize=12)
```

```
pylab.xlabel("t")
pylab.ylabel("Vo(t)")
pylab.grid(True)
```

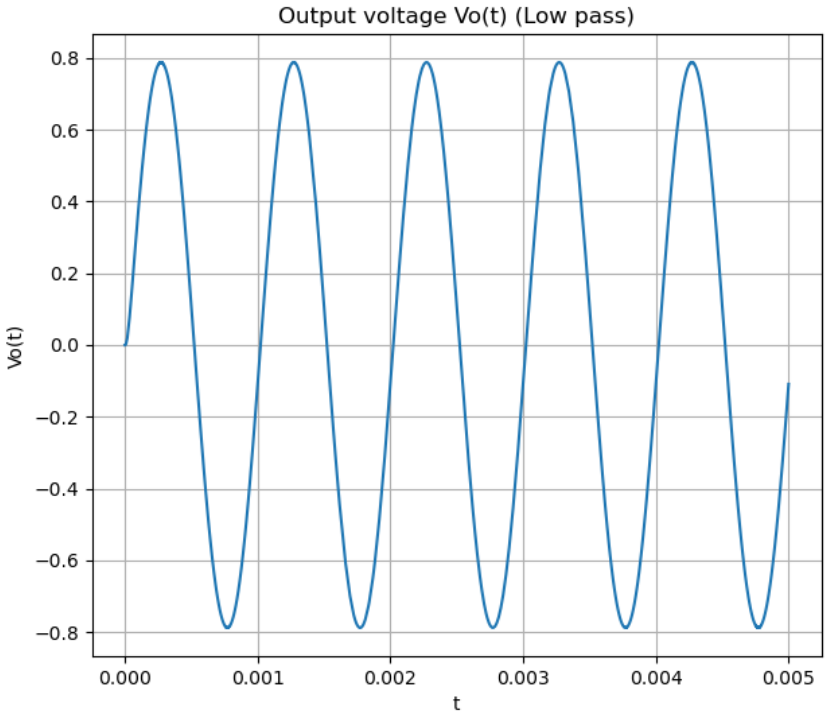

Figure 4: Output Voltage for sum of sinusoids

2.3 Response to damped sinusoid

In this case we assign the input voltage as a damped sinusoid like,
Low frequency,

$$V_i(t) = e^{-500t}(\ \cos(2000\pi t)\ )u_o(t)\ V$$

High frequency,

$$V_i(t) = e^{-500t}(\ \cos\big(2 * 10^6\pi t\big)\ )u_o(t)\ V$$

2.3.1 High Pass filter response

Below is the code for calculating output of High Pass filter for damped sin functions

```
def hp_sinusoid(b):
    A3, b3, V3 = highpass(10000, 10000, 1e-9, 1e-9, 1.586, 1)
    Vo = V3[3]
    H3 = sympytolti(Vo)

    a = -500
    b = b * PI

    t = np.linspace(0, 1e-2, 100000)
    Vin = np.exp(a * t) * np.cos(b * t)
    t, Vt, svec = sp.lsim(H3, Vin, t)

    return (t, Vt, svec)
```

Output for Low and High frequency inputs

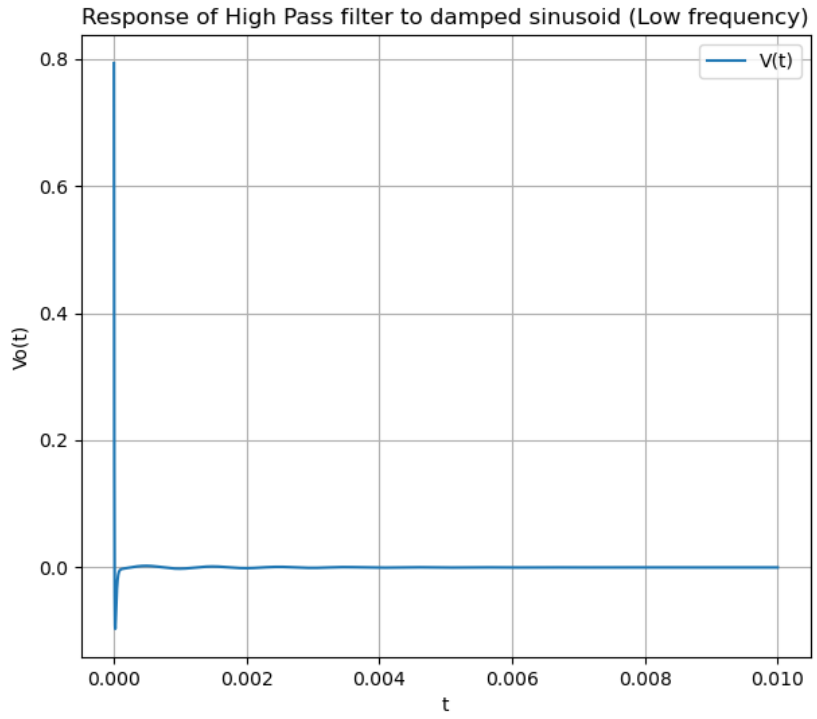

Figure 5: Output for Low frequency sinusoid

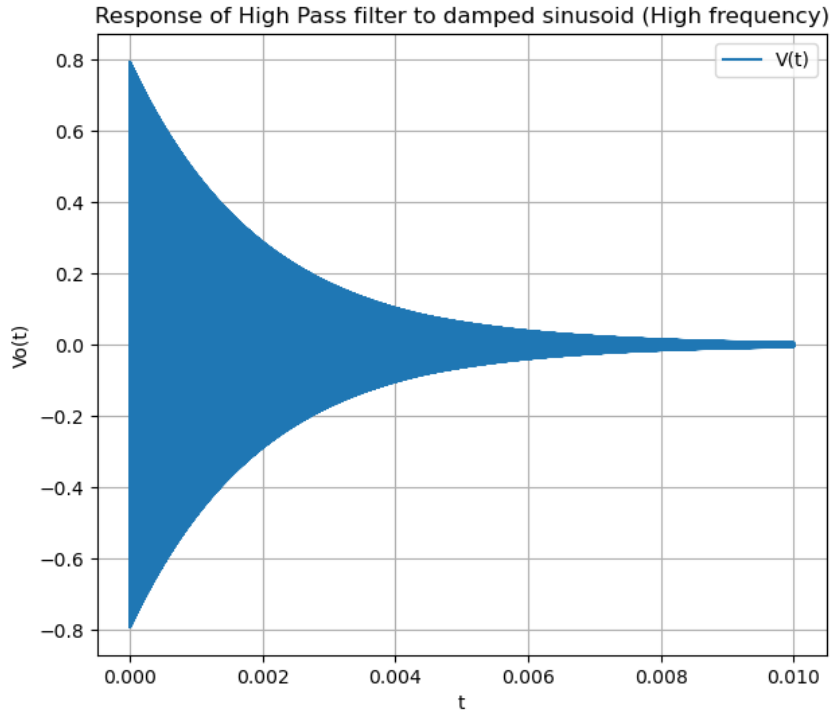

Figure 6: Output for High frequency sinusoid

2.3.2 Low Pass filter response

Below is the code for calculating output of Low Pass filter for damped sin functions

```
# Calculate response of Low Pass Filter to sinusoid
def lp_sinusoid(b):
    A4, b4, V4 = lowpass(10000, 10000, 1e-9, 1e-9, 1.586, 1)
    Vo = V4[3]
    H4 = sympytolti(Vo)

    a = -500
    b = b * PI

    t = np.linspace(0, 1e-2, 100000)
    Vin = np.exp(a * t) * np.cos(b * t)
    t, Vt, svec = sp.lsim(H4, Vin, t)

    return (t, Vt, svec)
```

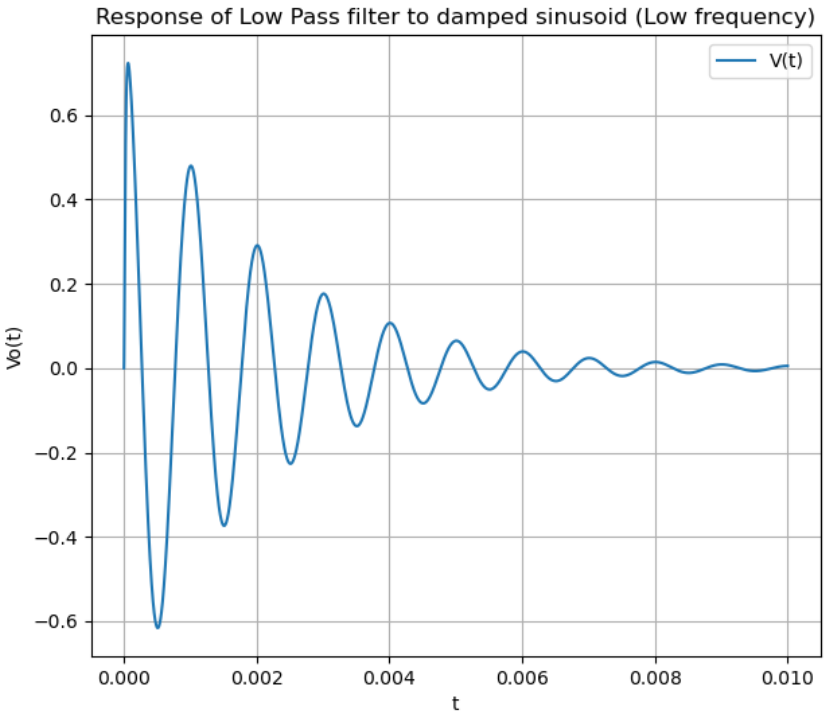

Figure 7: Output for Low frequency sinusoid

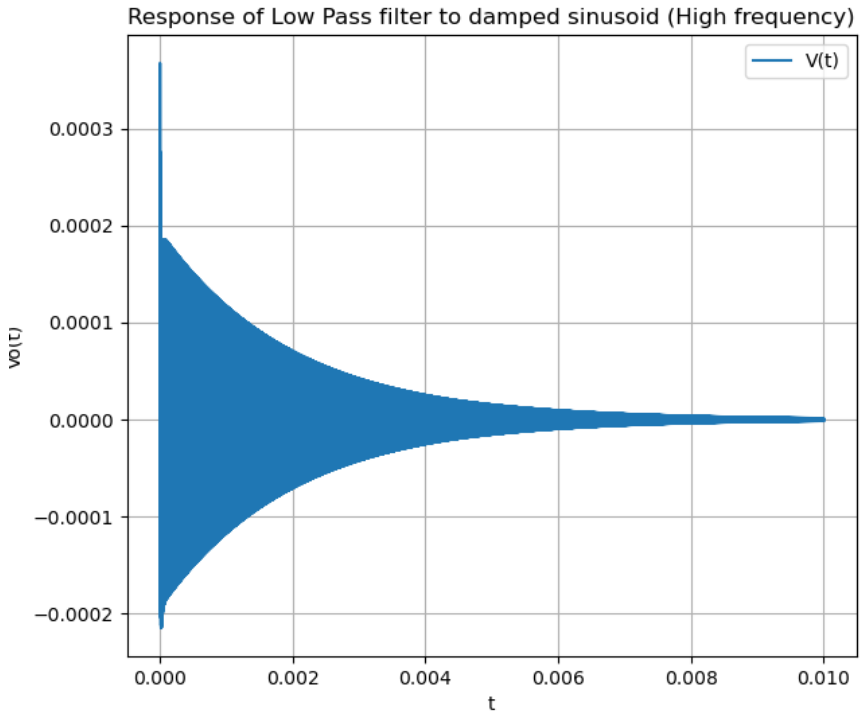

Figure 8: Output for High frequency sinusoid

2.4 Step Response of High Pass filter

Step response can be calculated by substituting, $Vi(s) = 1/s$.

```
# Step response of a High pass filter
A3, b3, H3 = highpass(10000, 10000, 1e-9, 1e-9, 1.586, 1 / s)
Vo = H3[3]
Vh = sympytolti(Vo)

t, Vt = sp.impulse(Vh, None, np.linspace(0, 5e-3, 10000))
pylab.figure(9, figsize=(7, 6))
pylab.plot(t, Vt, label="V(t)")
pylab.title("Step response of High pass filter", fontsize=12)
pylab.xlabel("t", fontsize=10)
pylab.ylabel("Vo(t)", fontsize=10)
pylab.grid()
pylab.legend()
```

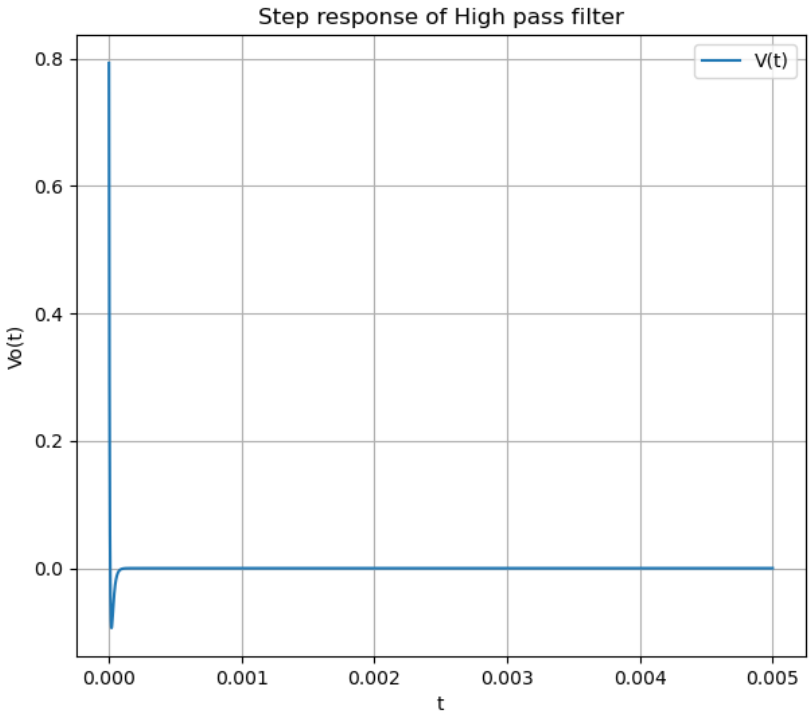

Figure 9: Output for High frequency sinusoid

3 Conclusion

The sympy module has allowed us to analyse circuits by analytically solving their node equations. And we used signal toolbox in python to find the time domain responses of the frequency responses